Design and Deployment of a National-Scale Authentication Infrastructure

Randy Butler[*], Doug Engert[†], Ian Foster[‡§], Carl Kesselman[¶], Steven Tuecke[‡], John Volmer[†], Von Welch[*]

Abstract

Increasingly, independent institutions with similar goals and interests are forming loosely coupled *virtual organizations* for collaboration and resource sharing. The construction of virtual organizations is hampered, however, by two conflicting goals: all members of the organization should have access to a resource as if it was their own, but participating institutions must not be required to change local security mechanisms or surrender control over their access control policies. We describe our experience designing, developing, and deploying the Grid Security Infrastructure (GSI), an authentication and authorization infrastructure that meets these requirements. GSI capabilities include single sign-on, no plaintext passwords, proxy credentials, mapping to local security mechanisms (including Kerberos), site control over access control policies, and user-controlled delegation. We have deployed GSI in the NSF-funded Partnerships in Advanced Computational Infrastructure, a national-scale virtual organization that comprises major research universities and laboratories. GSI-based versions of popular utilities including ssh and ftp are being used to provide access to dozens of supercomputers and storage systems nationwide.

1 Introduction

It is increasingly common in science and industry that individuals and institutions form *virtual organizations* that pool resources to tackle a common goal. One such example is the two Partnerships for Advanced Computational Infrastructure (PACIs), formed to create next-generation infrastructures for computational science. The PACIs are relatively large and long-lived virtual organizations: they are funded by the National Science Foundation (NSF) for 5–10 years and together link some 50 institutions and thousands of researchers. Other virtual organizations—for example, the fleeting one formed to write this article—are more limited in timescale and size.

Participants in virtual organizations commonly need to share resources such as data archives, computer cycles, and networks. In all but the most trivial cases, resources are not offered unconditionally; rather, restrictions are placed on their use, based on the identity of the user and the nature of the resource use requested. Thus, underlying any sharing mechanism is the ability to *authenticate* the identity of the requestor of a resource, and to determine if the requestor is *authorized* to make the resource request.

Because virtual organizations tend to be fluid, the mechanisms used to establish identity and authorization must be flexible and lightweight, so that resource-sharing arrangements can be established and changed quickly. However, because virtual organizations complement rather than replace existing institutions, sharing mechanisms cannot require changes to local policies and must allow individual institutions to maintain ultimate control over their own resources.

In this article, we describe our experiences creating and deploying an authentication and authorization infrastructure that meets these requirements. This infrastructure is based on the Grid Security Infrastructure (GSI) [3] developed within the Globus research project [1] and provides a secure, *single sign-on* capability, while preserving site control over access control policies and local security infrastructure. The infrastructure provides GSI-enabled versions of common applications, such as ftp and remote login, as well as a programming interface for constructing secure applications. The infrastructure is being used on a daily basis to access dozens of supercomputers and storage systems nationwide; it is one of a very small number of security solutions with this level of acceptance.

[*]National Center for Supercomputing Applications, Urbana, IL

[†]Electronic and Computing Technologies Division, Argonne National Laboratory, Argonne, IL 60439

[‡]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

[§]Department of Computer Science, The University of Chicago, Chicago, IL 60637

[¶]USC Information Sciences Institute, Marina del Rey, CA 90292

The rest of this article combines elements of a tutorial and case study, first describing the PACI environment and its security requirements, then reviewing the Grid Security Infrastructure, and finally presenting our experiences deploying GSI in this large virtual organization.

2 Multisite Authentication

Providing a means for reliably determining the identity of a requestor is critical to the construction of virtual organizations. However, authentication across multiple sites is complicated by the independent nature of the participants. We illustrate this point by describing some of the issues that arise in a specific virtual organization: the NSF PACIs. We then discuss the technical requirements for a multisite authentication infrastructure, and we review technology alternatives.

2.1 The PACI Community

The NSF PACIs are two distinct consortia that are collectively dedicated to the development of next-generation scientific problem-solving tools. Together, they total some 50 or more universities and government laboratories. As such, they are both large virtual organizations in their own right and providers of resources to an even larger (and less formal) national user community of many thousands of researchers, educators, and students located within universities, laboratories, and industry. While we focus our discussion on the PACIs, the issues raised are applicable to virtual organizations in general.

Various subsets of this PACI community interact in different ways, for example to develop a next-generation software system, to operate a remote electron microscope, or to access supercomputers. Most of these interactions involve some form of authentication and authorization.

PACI institutions all have long histories in running computing facilities. Each has well-established policies and procedures governing various aspects of facility operation, including computer security and, in particular, authentication and authorization. While several member sites run various flavors of Kerberos and DCE, many others run standard Unix authentication (DES-encrypted passwords, living in /etc/passwd or NIS). A few participating sites use one-time password mechanisms.

In the early days of the PACIs, there was some movement to convince a core set of sites to run Kerberos, with the vague hope of eventually establishing Kerberos cross-realm authentication between sites. However, this approach quickly proved infeasible for various technical, financial, and political reasons. It is now clear that member institutions will continue to use both Kerberos-based and non-Kerberos solutions within their sites for the foreseeable future. Resource sharing mechanisms used within the PACI community must be able to coexist with these different local mechanisms.

2.2 Technical Requirements

In the absence of any integrated authentication and authorization solution, virtual organizations have used a variety of ad hoc solutions to achieve resource sharing. Individual users typically have numerous accounts at the various institutions, with distinct login names and passwords at each institution. Information that needs to be shared might be placed on Web sites with access controlled via other passwords. This multiplicity of mechanisms and passwords represents a severe administrative headache for users and in practice discourages information sharing and collaboration. It also hinders the ability to build software that securely spans resources at multiple institutions or that allows secure collaboration between users at multiple institutions.

In designing an authentication and authorization solution that addresses the needs of virtual organizations, we must consider the requirements of both resource users and the sites providing resources.

User Requirements. From the user perspective, the primary requirement is simplicity: access to resources in the virtual organization should not look significantly different from access to resources in the user's own organization. Users should be able to "log on" (authenticate) just once and then have access to any resource in the virtual organization that they are authorized to use, without further intervention (often called *single sign-on*). Programs running on a user's behalf should be able to access the permitted resources as well, endowing these programs with a subset of the user's rights (i.e., delegation).

Consequently, any proposed solution must interface transparently to the tools that are commonly used to effect remote resource access: remote login (telnet/rlogin), file access (FTP), Web browsers, and programming libraries, such as CORBA or MPI. It must also be possible to develop new intersite applications, by providing standardized APIs for accessing security functions. For example, a group developing collaborative design tools should be able to integrate authentication and authorization mechanisms easily.

Site Requirements. The concerns of resource-providing sites tend to constrain the design of an authentication and authorization infrastructure in two ways. First, sites are typically *not* prepared to replace or modify whatever intradomain security solutions are already in place. Hence, a distinct interdomain security solution is required; this solution must interoperate with the local security solutions, must be at least as strong as those local solutions (so that its use does not weaken site security), and must be easily understandable by site administrators (so that they can trust it). Second, site administrators must be able to exert tight control over the policies that govern access to their resources, preserving control over both how users establishes their identity and which users are allowed to access which resources.

2.3 Technical Alternatives

We review briefly two widely used authentication approaches, Kerberos and secure shell, and explain why they do not meet our requirements.

Kerberos. Kerberos [7] is an authentication system used either in its own right or in the context of DCE. Kerberos allows users to authenticate via a secure transaction with a centrally maintained key server. Interorganizational authentication (called cross-realm authentication) can be achieved by designating key servers in other organizations whose authentication decisions one is willing to trust.

While Kerberos meets many of the basic requirements for virtual organization authentication, it presents two problems. First, using Kerberos for intersite authentication would require that it also be used for *intrasite* authentication—which is simply not feasible. Second is the practical difficulty encountered in negotiating cross-realm authentication agreements: sites often feel that they surrender too much control over local policy when they agree to accept tickets issued by other sites. In addition, the number of such agreements that must be negotiated is large.

Secure Shell. Secure shell (SSH) is a widely used remote login technology that meets a number of our requirements: It is based on public key authentication technology, uses link encryption to protect user credentials, and is easily deployed. Users like SSH because it provides basic remote login and file copy capabilities without a lot of complexity.

SSH also has two significant drawbacks as an authentication solution for virtual organizations. First, it requires that users manage their own cross-site authentication relationships, by copying public keys (or keeping track of passwords) to each site for which remote access is required. This task can become burdensome if users wish to access many remote resources; moreover, SSH denies sites control over authorization, making it difficult for example to deny access to a particular user without being invasive of user privacy. A second disadvantage is that SSH supports only a limited set of capabilities (remote shell and file transfer). Other applications (e.g., a collaborative environment or Web browser) that require authentication cannot benefit from SSH.

3 Grid Security Infrastructure

Recognizing the shortcomings of current authentication systems, we have developed the Grid Security Infrastructure as an alternative approach to intersite security. GSI was initially developed within the Globus research project [1] to support distributed computing environments, or computational grids [2], which have many features in common with virtual organizations. In the rest of this section, we provide a brief overview of GSI as described in [3]. As we explain in the next section, deployment and production use in the PACI framework have motivated a number of extensions.

GSI is concerned with *interdomain* operations: it bridges the gap between the different local security solutions at a virtual organization's constituent sites. The significant features of GSI (also summarized in Figure 1) are as follows:

- Each entity (user, resource, program, etc.) is assigned a globally unique name, or identity. We represent identity by a *certificate*, which specifies the name and additional information that can be used to identify the entity (e.g., a public key). In GSI, we represent certificates using the standard X.509 format. A *certification authority*, or CA, is a trusted third party that is responsible for assigning an identity to a name.

- Each entity is also provided with a means of proving that it possesses a specific identity. In the basic GSI implementation, identity checking is implemented by the authentication algorithm defined by the SSLv3 protocol. The veracity of the entity's identity is only as good as the trust placed in the CA that issued the certificate in the first place. Thus the authentication algorithm must validate the identity of the CA as part of the authentication protocol.

- An entity may delegate a subset of its rights to a third party (such as a process created by a program) by creating a temporary identity called a *proxy*. A proxy certificate is a certificate signed by the user or a previous proxy for the user, thus creating a chain of signatures terminating with the CA that issued the initial certificate. By checking the certificate chain, processes started on separate sites by the same user can authenticate to one another without requiring users to send their credentials to either of the sites.

- Each resource is allowed to specify the policy used to determine whether to accept incoming requests. In the initial GSI, this policy was expressed by a simple access control list; other techniques are used now (see Section 4).
- While the authentication protocol verifies the global identity of the parties involved, this name must be converted to a local subject name (e.g., a login name or Kerberos principal) before it can be used by the local security system. The GSI performs this operation by consulting a simple text-based *map file* that defines the binding between a global name and a local name. This map file is under the control of the local site.
- Access to security operations is provided by the GSS-API [5], a standard interface for expressing security operations. Our implementation of the GSS-API uses Secure Socket Layer Version 3, (SSLv3) as implemented by the SSLeay package, for its authentication protocols and supports the use of proxy certificates. SSLv3 is widely used for Web security, has been well scrutinized for security problems, and has broad acceptance as a mature protocol.

While relatively simple, this architecture meets all of the user and system requirements identified above. From the user's perspective, the global name and use of proxy credentials means that the user need authenticate only once prior to accessing any resource; proxy credentials and delegation also allow programs running on a user's behalf to access resources. The use of the X.509, SSLv3, and GSS-API standards means that it is straightforward to develop GSI-enabled versions of both common tools and more complex applications.

From a site's perspective, the architecture has the advantage of not requiring any change to local security infrastructure. Instead, the site needs to install just relatively simple GSI-enabled servers, which use well-known standards. Site control over policy is provided by the access control list/map file. Site administrators thus feel comfortable with the code and are prepared to deploy it alongside SSH and other remote access mechanisms.

4 GSI Extensions

GSI was implemented as part of the Globus toolkit and has been successfully deployed and used at over eighty sites. However, in moving GSI from the research environment into a production system, a number of extensions to GSI had to be made to address the operational concerns of production facilities. Of these the most significant were support for multiple certification authorities, interface to local Kerberos environment, and support for smartcards.

4.1 Multiple Certification Authorities

In the initial GSI implementation, we made the simplifying assumption that all user credentials were associated with a single CA operated by the Globus project. In practice, users need to be able to present credentials obtained from any source: their site's CA, a CA associated with a virtual organization such as the PACIs, or a commercial CA. Hence, GSI must allow a site to deal with credentials whose authenticity is attested to by different CAs. At the same time, a site must retain control, as part of its access control policy, over what CAs it is prepared to trust and what it will trust those CAs to do.

Applications such as web browsers often provide control over which CAs are trusted (e.g., by maintaining a list of "trusted" CA certificates) but do not provide control of what these CAs are trusted to do. Yet in practice we may wish to trust only some certificates signed by a given CA. For example, a site might trust a CA set up at a high school to sign certificates for its students (who would then be allowed access to educational materials), but would probably not want to trust certificates for other users signed by that CA.

We have extended the GSI implementation to address these issues by calling a general access control function, specified using the Generic Authorization and Access Control (GAA) API [6]. This function enables an application to reject an authentication operation based on the subject name and the entire chain of certificate signers, rather than simply the identity of the issuing certificate authority. Hence, users can present credentials obtained from any CA, and a site can decide whether or not to accept them.

Because of the policy differences that inevitably arise between sites regarding which CAs are accepted, users sometimes need to supply multiple credentials if they are to use resources at multiple sites. This requirement raises complex issues, which we are still investigating. To support single sign-on, users may require different credentials at different times, which makes delegation more complicated since multiple

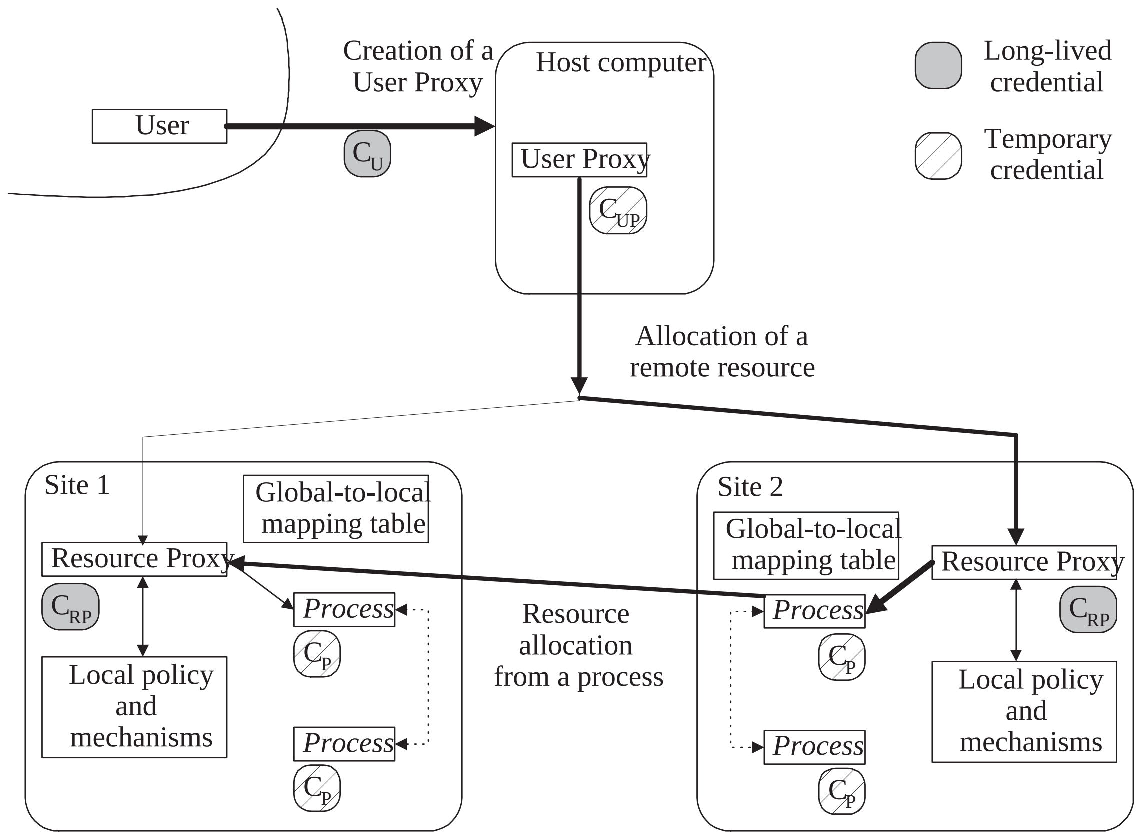

Figure 1: Schematic of the Grid Security Infrastructure, showing the basic operations supported: creation of temporary proxy credentials; authenticated requests to remote resources ("resource proxies"); authorization and global-to-local identity mapping; creation of remote processes, which may themselves issue further requests; and authenticated interprocess communication (dashed lines).

credentials may need to be delegated. When accessing a particular resource, a multiply credentialed user must determine which credential to supply: a protocol for determining this information is preferable to hit-and-miss guessing. Finally, the protocol used for mutual authentication between processes running on a user's behalf also needs enhancement, since the processes might be authenticated using different credentials.

As part of the PACI-wide deployment of GSI, one of the two PACIs, the National Computational Science Alliance, has created a CA for use by sites that do not want to run their own. The creation of this CA has been a significant effort in its own right, because of the need to define a certificate policy that was acceptable to participating institutions. This policy builds on the model certificate policy developed by the Federal Public Key Infrastructure project.

4.2 Obtaining Kerberos Credentials

In the simplest case, interfacing to a local security environment requires that the GSI map from the global name to a local subject name based on an entry in a map file. in practice, however, many computing sites utilize an intrasite security solution based on Kerberos technology. Thus, we must also obtain a set of local credentials in the form of Kerberos tickets.

To obtain such a ticket, GSI must be able to authenticate to a Kerberos realm, DCE cell or AFS cell, and it must be possible for Kerberos to issue tickets based on GSI authentication including proxies. To facilitate this, we have developed a modified Kerberos Key Distribution Center called SSLK5D, The ticket returned by SSLK5D can be used like any forwarded ticket generated by Kerberos. Control of the Kerberos realm or DCE cell still remains with the security administrator, who controls the SSLK5D and its database to map certificate names to Kerberos principals.

SSLK5 is similar in function to the IETF's PK_INIT draft standards and DCE RFC 68.4, but with one distinction: we use SSL to communicate to the SSLK5D server, rather than a specialized protocol. However, as PK_INIT and RFC 68.4 become part of the standard environment, they can replace SSLK5.

As illustrated in Figure 2, this support for the creation of Kerberos credentials means that GSI is able to interoperate with, rather than replace, DCE.

4.3 Alternative Credential Management

Our GSI implementation, like many other public key systems, maintains the user's private key in the user's local file system, in encrypted form. During the sign-on process, the user provides a pass phrase to decrypt the private key. This approach has three disadvantages. First, the private key may not be accessible when the user travels to a remote location. Second, a careless user may expose the pass phrase by authenticating from a networked terminal. Third, an adversary may retrieve the encrypted private key and then subject it to a passphrase-guessing attack.

Because these issues were of concern at some GSI sites, we have extended GSI to allow the user's private key to be stored on a *smart card*, a credit card-sized device containing a small amount of memory (typically 4-16 KB with today's technology) and a microprocessor capable of performing 1024-bit cryptographic signing operations. With this technology, a user's private key never leaves the card: during logon, the proxy generation code negotiates with the card to obtain a signed proxy certificate.

5 Building the Infrastructure

Besides the augmented GSI, various elements are needed to construct an infrastructure that meets the requirements of Section 2.2. These include (1) a set of *GSI-enabled applications*, (2) various *GSI-enabled toolkits*, and (3) a certificate authority policy and PACI certificate authority. We discuss each of these activities as well as other issues that have arisen while deploying this infrastructure.

5.1 GSI-based Applications

True single sign-on means that PACI users should be able to use their GSI credentials for *all* PACI authentication purposes as well as more sophisticated applications such as distributed supercomputing and collaborative data analysis. With this goal in mind, we have developed GSI-enabled versions of a variety of common applications, and have also verified that our credentials can be accepted by commercial Web browsers.

A GSI-enabled version of SSH was created by modifying SSH to use GSSAPI as one of its authentication mechanisms. This change allows users to use their GSI credentials for authentication to a GSI-enabled `sshd` daemon, including doing delegation. Since SSH supports multiple mechanisms (e.g., Kerberos, RSA key pair, password), it can support the GSI and still

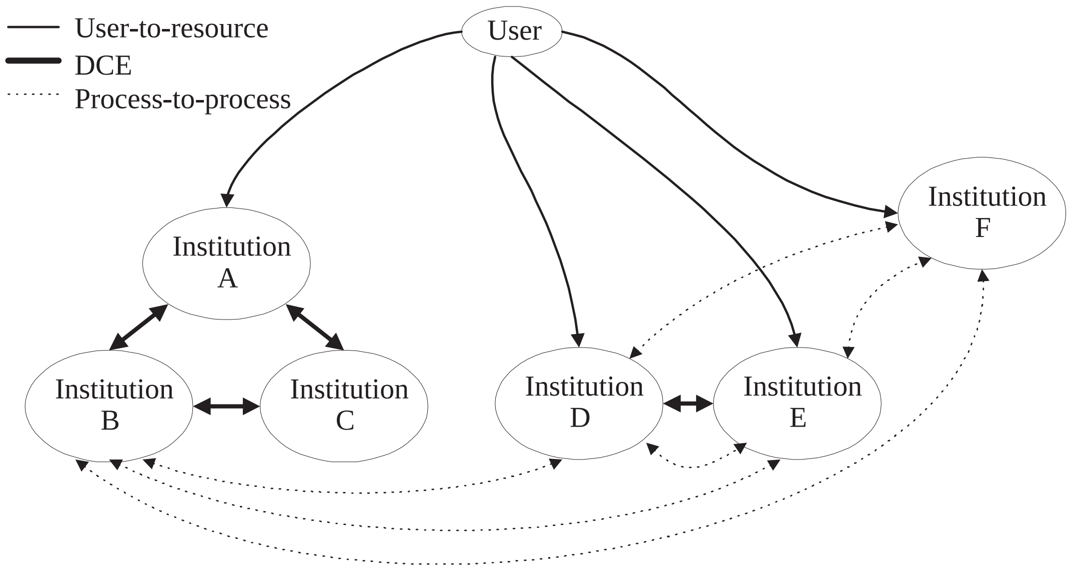

Figure 2: GSI in use in an environment in which some sites (A, B, and C, and D and E) use DCE for inter-site authentication. GSI allows a user to access resources in multiple such "DCE clouds," while also allowing for DCE to be used between sites when it is available.

be fully backwards compatible. The SSH modifications were also implemented by Van Dyke Technologies in their commercial SSH product, SecureCRT. Adding this support to SSH was a relatively easy task requiring only a couple of weeks.

Our use of GSS-API also makes development of a GSI-enabled ftp client and server straightforward. The Kerberos project at MIT already had a ftp modified to work with the Kerberos 5 GSSAPI library based on RFC 2228. This code required only minor changes in order to work with the GSI SSLeay library. NCSA's mass storage system also uses a ftp interface for user access, which had already been modified to allow authentication via Kerberos using its GSSAPI library. Again, little effort was required to allow this system to accept GSI credentials.

We have also verified that commercial Web browsers can function with certificates generated for GSI use. This work required no actual changes to the browsers themselves, just experimentation to determine the proper procedures for formatting and importing the certificates into the browsers.

5.2 GSI-based Toolkits

While the set of GSI-enabled applications just described provide a good basis for remote resource access, members of a virtual organization such as the PACIs also want security to be incorporated into a wide variety of other applications. Hence, we have developed various tools that facilitate the incorporation of GSI mechanisms into applications.

One such tool is `gss_assist`, a set of convenience functions for accessing GSS functions. GSS-API is rich and robust but is also complex. Many applications require only a subset of GSS-API features; `gss_assist` shields the developers of these applications from unneeded complexity, by providing a simpler API that still implements the full security of GSI. Many `gss_assist` functions are simple wrappers around their GSSAPI counterparts, with appropriate default values. Some are more complicated, such as the wrappers around the security context initialization and acceptance functions that perform the full looping and network communication needed for GSS-API authentication and context establishment.

Another useful GSI-based application toolkit is Globus, a set of services for constructing distributed applications. Globus mechanisms have been extended to use GSI functionality; hence, any application that uses Globus mechanisms gets security essentially for free. For example, a version of the popular Message Passing Interface, MPICH-G, uses Globus mechanisms for initiating remote computation and hence does not need to do anything special to address authentication and authorization issues when running across multiple sites.

6 Conclusions

We have described our experiences developing and deploying the Grid Security Infrastructure for a large virtual organization, namely, the NSF Partnerships for Advanced Computational Infrastructure. GSI capabilities include single sign-on capability, standards-based APIs and applications, and delegation mechanisms, all without requiring changes to the security mechanisms or policies at a particular site. The Grid Security Infrastructure is being used to support remote access to supercomputers, data archives, and other resources at dozens of sites across the United States.

Acknowledgments

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38; by the Defense Advanced Research Projects Agency under contract N66001-96-C-8523; by the National Science Foundation; and by the NASA Information Power Grid program.

References

[1] I. Foster and C. Kesselman. Globus: A toolkit-based grid architecture. In *[2]*, pages 259–278.

[2] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure.* Morgan Kaufmann Publishers, 1999.

[3] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM Press, 1998.

[4] W. Johnston and C. Larsen. A use-condition centered approach to authenticated global capabilities: Security architectures for large-scale distributed collaboratory environments. Technical Report 3885, LBNL, 1996.

[5] J. Linn. Generic security service application program interface. *Internet RFC 1508*, 1993.

[6] Tatyana Ryutov and Clifford Neuman. Access control framework for distributed applications. Internet Draft, Internet Engineering Task Force, November 1998.

[7] J. Steiner, B. C. Neuman, and J. Schiller. Kerberos: An authentication system for open network systems. In *Usenix Conference Proceedings*, pages 191–202. 1988.